ECAI 2016
G.A. Kaminka et al. (Eds.)

doi:10.3233/978-1-61499-672-9-347

A Reinforcement Learning Framework for Trajectory Prediction Under Uncertainty and Budget Constraint

Truc Viet Le[1] and **Siyuan Liu**[2] and **Hoong Chuin Lau**[3]

Abstract. We consider the problem of trajectory prediction, where a trajectory is an *ordered* sequence of location visits and corresponding timestamps. The problem arises when an agent makes sequential decisions to visit a set of spatial locations of interest. Each location bears a stochastic utility and the agent has a limited budget to spend. Given the agent's observed *partial* trajectory, our goal is to predict the agent's remaining trajectory. We propose a solution framework to the problem that incorporates both the stochastic utility of each location and the budget constraint. We first cluster the agents into groups of homogeneous behaviors called "agent types". Depending on its type, each agent's trajectory is then transformed into a discrete-state sequence representation. Based on such representations, we use reinforcement learning (RL) to model the underlying decision processes and inverse RL to learn the utility distributions of the spatial locations. We finally propose two decision models to make predictions: one is based on long-term optimal planning of RL and another uses myopic heuristics. We apply the framework to predict real-world human trajectories collected in a large theme park and are able to explain the underlying processes of the observed actions.

1 Introduction

How does a rational agent decide to visit a set of locations in space? Assuming there are distinct points of interest (POIs), then the act of visiting them has to happen sequentially. We call it *spatial* sequential decision-making. It is reasonable to assume that each location bears a non-negative utility (reward) to the decision-maker that would not be fully realized until it is visited. Until then, utilities remain *uncertain* and reflect the agent's prior preferences. When making sequential decisions, a rational agent should also weigh in the long-term costs of visiting each of the locations in order to make an optimal plan, where "costs" here are assumed to be proportional to physical distances. Hence, answering the question above would require a model of the agent's sequential decisions for selecting locations, whose utilities remain uncertain and costs are dynamic, and weighing in their long-term consequences into the decision-making [15].

In practice, the agent typically has a limited amount of resources (e.g., time) to run its plan, which we call a *budget*. Such a budget constraint can significantly shape the agent's decision-making process and outcomes in non-obvious ways. In this paper, we propose a framework based on reinforcement learning [24] to model the agent's spatial sequential decision-making, taking into account the uncertainty of the utilities and the budget constraint. Using the framework, we could discover the underlying processes that drive real-world behaviors such as the condition for making long-term optimal decisions in the defined setting. Indeed, traditional economic view of rational decision-making as solving an optimization problem often fails to predict reality due to *bounded rationality* [10]. Such discoveries would give insights into real-world human behaviors and help bridge the gap between human and machine intelligence [28, 15].

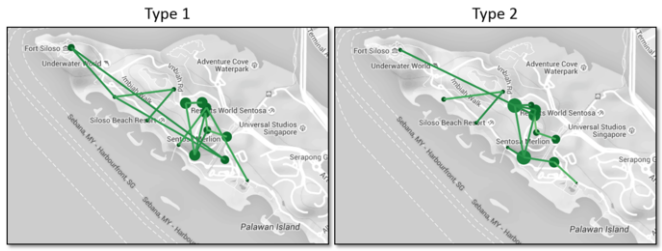


Figure 1: Visualizing the attractiveness of the same set of POIs in a real-world theme park environment (to be discussed in Sect. 7) and the pairwise transition probabilities (only those probabilities ≥ 0.20 are drawn) between them as observed by two groups of agents: "Type 1" and "Type 2". Each group is given a certain amount of time budget to visit the set of POIs, where Type 1 has, on average, 114 minutes more than Type 2. The size of each POI is drawn to reflect its relative popularity (attractiveness) among members of each group.

Our motivation comes from the problem of predicting the *next* sequence of location visits (called *trajectory*) of a mobile agent knowing its current trajectory and past observed trajectories of other similar agents. Accurate predictions of the agent's next locations can enable numerous applications of location-based services such as real-time prediction of visitor arrivals and congestion at POIs or devising real-time advertising strategies or adaptive recommendation system for a mobile agent knowing its probable future trajectory.

Consider the example illustrated in Fig. 1, whose data were collected from real-world human behaviors in a theme park (to be discussed in Sect. 7). In this setting, suppose there are two groups of agents (human visitors) of equivalent sizes called "type 1" and "type 2". Each agent in each group is to visit the same set of POIs within a given time frame (budget). Agent type 1 is given, on average, 114 minutes more than type 2. Such a budget difference can translate into starkly different behaviors as illustrated in the figure. Not only is the relative attractiveness of each of the POIs different, but the pairwise transition probabilities among them also become discernibly distinct. Type 1 appears to have a larger "coverage" of the POIs through their sequential transitions, while type 2 tends to visit those POIs that are

[1] Singapore Management University, 80 Stamford Rd., Singapore 178902. E-mail: trucviet.le.2012@smu.edu.sg
[2] Smeal College of Business, Pennsylvania State University, PA 16802, USA. E-mail: siyuan@psu.edu
[3] Singapore Management University, 80 Stamford Rd., Singapore 178902. E-mail: hclau@smu.edu.sg

clustered together. These observations reflect the inherently different underlying decision processes used by these agent types. Thus, in order to make accurate trajectory predictions, it suffices to model the sequential decision-making process of each group separately.

In this paper, we develop on and extend the capabilities of the framework proposed by Le *et al.* [17] for spatial decision modeling. Specifically, we set out to predict an *ordered* sequence of an agent's future locations (as opposed to an *unordered* bundle). Furthermore, our novel contribution is that we do not rely solely on a generative model as previously proposed to generate sequential actions (e.g., naive Bayes [14] or hidden Markov models (HMMs) [20]). Instead, we integrate one of such (i.e., HMMs) into a reinforcement learning framework to model an agent's sequential decisions. We further propose decision models based on the learned utilities resulted from the framework for trajectory prediction. Doing so enables us to explain the underlying processes of the predicted outcomes, the effects of budget constraint on decision-making, and evaluate the appropriateness of the proposed decision models.

We summarize our contributions as follows:

- We model the sequential decision process of an agent in an integrated framework to predict its trajectory;
- Our framework takes into account both the stochasticity of rewards and the budget constraint;
- We propose two decision models for prediction: one is based on long-term optimal planning of reinforcement learning and another uses myopic heuristics;
- We empirically evaluate our framework using real-world human trajectories with compelling results.

2 Related Work

Trajectory prediction. The problem of predicting the future location(s) of a mobile agent is not entirely new. Krumm and Horvitz [14] propose a naive Bayes model called *Predestination* to predict the final destination of a driving trip given its partially observed GPS trajectory. In most recent work, some form of Markov model is often used to learn the observed transitions and infer future locations. For example, Mathew *et al.* [20] use hidden Markov models (HMMs) to identify clusters of locations from raw GPS data and Gambs *et al.* [9] propose a mobility model based on Markov chains to incorporate knowledge of the previous n visited locations. We also employ HMMs in our framework, but in a radically different way: to represent the environment in which the agent interacts with. Recently, Le *et al.* [17] propose a framework based on revealed preference learning to predict *unordered* bundles of spatial locations given an agent's budget information. In this respect, our work expands on [17] for predicting *ordered* sequences of spatial decisions.

Sequential decisions. Modeling human sequential actions has been traditionally studied in the domain of human-computer interaction. For instance, mining sequential behaviors has been used to discover mobile users that share similar habits [19], or to imitate human behaviors in order to provide better automated care to the disabled and the elderly [11]. In this respect, modeling sequential decisions as Markov processes is commonly used to simplify the representation of the user's knowledge [26]. A common shortcoming among these work is the lack of modeling of the users' underlying decision processes in order to explain the discovered patterns.

Reinforcement learning. Understanding human behaviors requires finding the reward function that motivates the observed actions. Inverse reinforcement learning (IRL), first proposed by Russell [22], provides an elegant framework to identify the reward function being optimized by the agents given observations of their activities. Ng and Russell [21] propose the original algorithms to tackle the problem based on linear programming. Ever since, there has been a wealth of algorithms developed to solve IRL [25]. IRL has enjoyed diverse applications in automated control systems that try to imitate the behaviors of expert human users (a.k.a. "learning from demonstrations") such as learning how to drive [2], controlling helicopters [1], and predicting mouse movements [26]. In this respect, our framework integrates IRL to model the stochasticity of rewards.

3 Problem Statement

We consider a set $\mathcal{D}$ of agents and a finite set $\mathcal{G}$ ($|\mathcal{G}| = n$) of POIs (locations). Each agent $i \in \mathcal{D}$ has a utility vector u_i over each location $j \in \mathcal{G}$, where $u_{ij} \in \mathbb{R}_{\geq 0}$ is the utility of j to i. Agent i has a budget constraint B_i and wishes to visit a subset $s_i \subseteq \mathcal{G}$ such that $\sum_{j \in s_i} c_{ij} \leq B_i$, where c_{ij} is i's cost of visiting j. We denote s_i as agent i's *trajectory* that contains the *ordered* sequence of locations visited by i and the corresponding timestamps. Without loss of generality, we assume throughout that the costs and budget constraint are in terms of travel time and i makes a binary decision vector $s_i \in \{0,1\}^n$. Hence, c_{ij} is a *dynamic* cost for each j that depends on the previous location in the sequence. We additionally assume the proportionality between distance and travel time, where all distances considered in this paper are spatial Euclidean distance.

Suppose $\mathcal{D}$ can be divided into non-overlapping subsets called *agent types*, where each "type" implies homogeneous preferences and behaviors. Given an agent of a certain type, his partial trajectory (say the first k location visits) and the current budget, our goal is to predict the agent's remaining trajectory. The notion of agent type comes from the idea that modeling each individual agent is impractical. It is much more feasible to divide them into finite and disjoint *clusters* of similar preferences and behaviors. Thus, we also use the terms "cluster" and "(agent) type" interchangeably.

Predicting an agent's remaining trajectory requires sequential decision modeling under uncertainty and budget constraint. The uncertainty comes from the utility distributions of the remaining locations. While the relative attractiveness of the locations can be easily worked out using a simple frequency count, it is not straightforward how to learn their utility distributions from the observed trajectories and how to incorporate them into a sequential decision-making model.

4 Solution Overview

We propose an integrated framework to model and predict the next sequence of locations given an agent's observed partial trajectory and budget constraint. The framework consists of two components: learning and prediction. Fig. 2 illustrates the overall framework. Table 1 summarizes the notations used in this paper.

Learning. We first divide the agents in the training set $\mathcal{S}$ into K finite clusters, where each cluster Cl_j ($1 \leq j \leq K$) represents an agent type. K is typically chosen heuristically via some clustering coefficient (e.g., the silhouette index). Using the agents' observed features and the K clusters as class labels, we train a multi-class classifier (e.g., multinomial logistic regression). We also model the environment that the agents interact with as a finite set of states S, where each state $s \in S$ has a distinct vector of features $\mathbf{f}_s$. We use hidden Markov models (HMMs) to transform the observed trajectories into finite sequences of states. Such a representation can then be

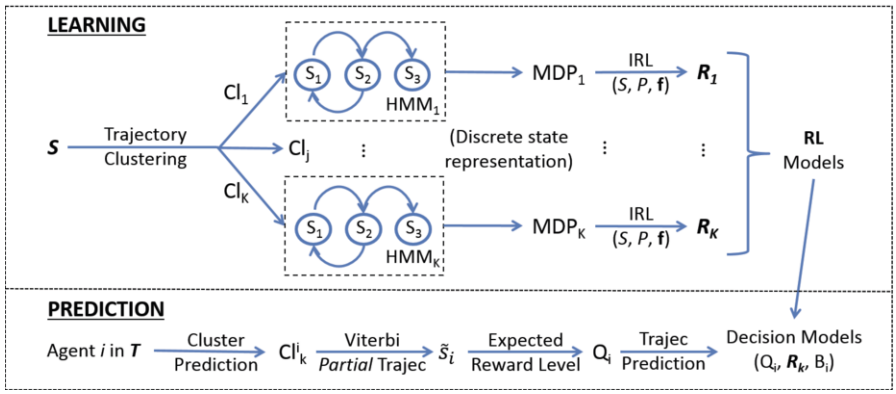


Figure 2: The proposed framework to model and predict the remaining trajectory of a test agent $i \in \mathcal{T}$ given its observed partial trajectory $\tilde{s}_i$, current budget B_i, and trajectories of the agents in the training set $\mathcal{S}$. "Trajec" stands for "trajectory". S is the finite set of states, P is the matrix of state transition probabilities, and $\mathbf{f}$ is the set of feature vectors of the states in S.

Table 1: Summary of notations used in the paper.

Notation	Description
$\mathcal{D}, \mathcal{S}, \mathcal{T}$	Total dataset, training set and test set, respectively
$\mathcal{G}$	Set of POIs, where $\|\mathcal{G}\| = n$
u_{ij}, c_{ij}	Utility and cost of location $j \in \mathcal{G}$ for agent i
s_i, $\tilde{s}_i$, B_i	Trajectory, partial trajectory, and current budget of i
l_i	Trajectory (sequence) length of agent i
K	Number of clusters (agent types)
S	Finite set of states for each agent type ($\|S\| = N$)
$\mathbf{f}_s$	Feature vector of each state $s \in S$
$\mathbf{R}_k$	State-reward matrix $\forall$ agent type k ($1 \le k \le K$)
$\mathbf{R}_a$	Location-reward matrix for each location $a \in \mathcal{G}$
Q_i	Expected reward level ("personal goal") of agent i

modeled as a Markov decision process (MDP). The utility of each action (i.e., location visit) can then be derived via the process of inverse reinforcement learning (IRL) using the agents' observed actions (represented in the transition probability matrix P of the MDP). The final outcomes of IRL are the reward matrices $\mathbf{R}$.

Prediction. Given the observed partial trajectory and features of an agent i in the test set $\mathcal{T}$, we first predict i's type Cl_k^i using the trained classifier above. We then use the Viterbi algorithm [7] to find the most probable sequence of states $\tilde{s}_i$ for the observed trajectory. We are then able to model i's goal Q_i (also called the "expected reward level") and predict the next sequence of visits that can meet this goal within budget B_i. We finally propose two decision models that take into account the uncertainty of the utilities (represented by the matrix $\mathbf{R}_k$ for each type k) and budget B_i.

In the following sections, we elaborate on each of the components of the proposed framework shown in Fig. 2.

5 Learning

5.1 Trajectory Clustering

For each agent $i \in \mathcal{S}$, let l_i ($1 \le l_i \le n$) be i's *sequence length*, which is the total number of locations visited by i. We denote the sequence of locations visited by i as $y^{(i)} = \{y_t^{(i)}\}_{t=1}^{l_i}$ and the sequence of corresponding timestamps as $\tau^{(i)} = \{\tau_t^{(i)}\}_{t=1}^{l_i}$. We define i's **trajectory** as $s^{(i)} = \{(y_t^{(i)}, \tau_t^{(i)})\}_{t=1}^{l_i}$. We are able to discretize $\tau^{(i)}$ into T segments, where Δ_τ is the duration of each segment. We can then derive a vector a_i of length T for each $s^{(i)}$, where each $a_{it} \in a_i$ ($1 \le t \le T$) indicates i's observed location at time t.

We can now cluster the agents based on the similarities a_i for all $i \in \mathcal{S}$ using, e.g., hierarchical clustering (because of its simplicity and effectiveness). In particular, we propose to use the agglomerative approach that clusters the vectors recursively from bottom up. To this end, we use the edit distance [6] to quantify the dissimilarity between a_i and a_j with the substitution cost being the distance between the pair of locations that differ. To select K, the hierarchical tree is "cut" at some height that splits $\mathcal{S}$ into K clusters. The goodness of the clustering can then be quantified using, e.g., the silhouette coefficient. We choose K that best aligns with our domain knowledge and produces a good enough clustering coefficient.

5.2 Environment Modeling

We use hidden Markov models (HMMs) to model the environment the agents interact with as a finite set of states $S = \{S_1, S_2, \ldots, S_N\}$. An HMM describes the relationship between an observed stochastic process and an unobserved (hidden) underlying process. The hidden process follows a Markov chain and the observations are conditionally independent given the sequence of hidden states. Let $\{Y_t\}_{t=1}^T$ and $\{X_t\}_{t=1}^T$ represent the observations and the corresponding hidden states, respectively. We denote $f(y_t|\Theta_{x_t}) = \Pr(Y_t = y_t; \Theta | X_t = x_t)$ as the (emission) density function of observation y_t parameterized over Θ given hidden state x_t. Each emission y_t is a tuple (y_k, τ_k) with the spatial component y_k being a discrete location drawn from $\mathcal{G}$ and the temporal component τ_k being a continuous timestamp drawn from the Gaussian distribution $\mathcal{N}(\mu_k, \sigma_k)$ ($1 \le k \le N$).

An HMM with N states is completely specified by:

1. The finite set of hidden states $S = \{S_1, S_2, \ldots, S_N\}$;
2. The state transition matrix $\mathbf{T} = \{t_{ij}\}$, where $t_{ij} = \Pr(X_t = S_j | X_{t-1} = S_i), 1 \le i, j \le N$;
3. The parameter vector Θ_i of the response (or emission) density function $f(y_t|\Theta_{x_t})$ for each S_i; and
4. The vector of initial (state) probabilities $p = \{p_i\}$, where $p_i = \Pr(X_1 = S_i)$ and $\sum_{i=1}^N p_i = 1$.

Each hidden state of the HMM can be thought of as a *spatiotemporal cluster* of the visiting activities. Empirical observations confirm that nearby locations are much more likely to be visited sequentially in short periods of time, i.e., having "high" emission probabilities. We fit the HMMs using the trajectories $s^{(i)}$ $\forall i \in \mathcal{S}$. A well-known method to estimate the parameters of an HMM is the Baum-Welch

algorithm [4]. For each HMM$_j$ ($1 \leq j \leq K$), we select the optimal number of states N_j^* using the Bayesian Information Criterion (BIC) [8]. An important inference problem is that given a sequence of observations, find the most probable sequence of hidden states that produces it, which can be solved using the Viterbi algorithm [7].

5.3 Inverse Reinforcement Learning

5.3.1 Preliminaries.

Markov decision processes (MDPs) [5] provide an elegant framework to model sequential decisions in an environment represented as a finite state space S. At each state $s \in S$, the agent chooses an action $a \in A$. Upon which, the process transitions into the next state $s' \in S$ according to the probability $P_a(s, s') = \Pr(S_{t+1} = s' | S_t = s, a_t = a)$. The agent then receives a reward $R_a(s, s')$. The main concern of MDP is to find an optimal policy $\pi^* : S \mapsto A$ that maximizes the long-term *cumulative* reward $\sum_t R_{a_t}(s_t, s_{t+1})$.

Let $P_{\pi(s)}$ represent the transition probability matrix corresponding to the application of some policy π. A finite-horizon MDP is completely described by the tuple $(S, A, P_{\pi(s)}, R)$. The value function $V^\pi(s)$ of policy π at state s represents the expected cumulative reward from s. Thus, our goal is to find an optimal policy π^* such that $V^{\pi^*}(s)$ is maximized. It can be shown that there exists at least one optimal policy such that $V^\pi(s)$ is maximized for all $s \in S$ [24] that can be expressed as:

$$\pi^*(s) \in \arg\max_{a \in A} \sum_{s' \in S} P_a(s, s')[R(s, s') + \gamma V^\pi(s')]. \quad (1)$$

A fundamental property of the value function is, for any policy π and any state s:

$$V^\pi(s) = R_{\pi(s)}(s) + \sum_{s' \in S} P_{\pi(s)}(s, s') V^\pi(s'). \quad (2)$$

Eqn. (2) (called the Bellman equation) directly gives rise to efficient dynamic programming (DP) formulations to find a long-term optimal policy π^* called value iteration and policy iteration [5].

Inverse reinforcement learning (IRL) is the inverse problem to MDP, whose goal is to determine the reward function R that is being optimized given observations of the sequential decisions. Ng and Russell [21] originally propose LP formulations to solve the problem with constraints leading to the optimal observed policy. Abbeel and Ng [2] later propose a strategy of *matching feature expectations* between an observed policy and an agent's behaviors. The strategy is both necessary and sufficient to achieve the same performance as if the agent were in fact solving an MDP with reward function linear in the features of the states. Denote ξ_i a state-based trajectory (aka a "path"), $\mathbf{f}$ the sequence of feature vectors of a path, and $\bar{\mathbf{f}} = \frac{1}{m} \sum_i \mathbf{f}_{\xi_i}$ the empirical expected feature count based on m trajectories. Matching feature expectations is described by:

$$\sum_{\xi_i} \Pr(\xi_i) \mathbf{f}_{\xi_i} = \bar{\mathbf{f}}. \quad (3)$$

We adopt the maximum entropy (MaxEnt) IRL algorithm [27] to learn the reward distribution of each state. MaxEnt IRL is an effective framework for modeling and understanding human activities, where the recovered reward function intuitively encodes an individual's set of preferences [12]. The notion of reward distribution comes from the fact that different people, even if classified into types, would still have different preferences (utilities) for the same thing. Such diversity in tastes can be best modeled as a probability distribution.

5.3.2 Maximum Entropy IRL (MaxEnt IRL).

Given a state-action sequence $\xi = \{(s, a)\}_i$, where $s_i \in S$ and $a_i \in A$, agent i is optimizing some function that linearly maps the features of each state $\mathbf{f}_{s_j} \in \mathbb{R}^k$ to a reward value that represents i's utility of visiting that state. This function is parameterized by some weight vector θ and the reward of a trajectory is simply the sum of all the state rewards. The reward weights are applied to the path feature counts $\mathbf{f}_\xi = \sum_{s_i \in \xi} \mathbf{f}_{s_j}$ such that the reward of the trajectory is the weighted sum of the feature counts along the path:

$$R(\mathbf{f}_\xi) = \theta \cdot \mathbf{f}_\xi = \sum_{s_j \in \xi} \theta \cdot \mathbf{f}_{s_j}. \quad (4)$$

Since many distributions of paths may match the feature counts and any one distribution from among this set may exhibit a preference for some of the paths over others not implied by the path features. Such ambiguity is solved using the principle of *maximum entropy* by choosing the distribution that does not exhibit any additional preferences beyond matching feature expectations. The resulting distribution over the paths is parameterized by the weights θ:

$$\Pr(\xi_i | \theta) = \frac{1}{Z(\theta)} e^{\theta \cdot \mathbf{f}_{\xi_i}} = \frac{1}{Z(\theta)} e^{\sum_{s_j \in \xi_i} \theta \cdot \mathbf{f}_{s_j}}, \quad (5)$$

where $Z(\theta)$ is some *partition function* for the parameter weights. This distribution also provides a *stochastic policy* (i.e., a distribution over the actions at each state). Refer to [27] for more details.

We now build an MDP model (S, A, P, R) for each agent type, where S is the set of states of the corresponding HMM and A is the set $\mathcal{G}$ of locations. We then need a set of state sequences in order to derive the transition matrix P and reward function R. To this end, we convert each trajectory into its most probable sequence of (hidden) states using the Viterbi algorithm [7]. P is then derived by sampling the observed state transitions and action taken at each state.

MaxEnt IRL additionally requires a set of features $\mathbf{f}_s$ for each state $s \in S$. We use the spatiotemporal characteristics of each state as its features. Specifically, recall that each state S_i of the HMM is both a spatial cluster (i.e., what locations are likely to be visited) and a temporal cluster (described by the Gaussian mean μ_i). We use the tuple $(lo_i, la_i, \mu_i, \sigma_i)$ as the features $\mathbf{f}_{S_i}$ of S_i, where lo_i and la_i are the "mean"[4] longitude and latitude coordinates of S_i and μ_i and σ_i are the mean and standard deviation of the Gaussian emission, respectively. Such weighted sum of the coordinates are referred to as the "cluster centroids" of the states. Hence, each state S_i admits a unique cluster centroid C_i described by its (lo_i, la_i).

Each run j of MaxEnt IRL produces a unique reward function $R_j : S_i \mapsto \mathbb{R}^+, \forall 1 \leq i \leq N$. In order to produce a *distribution* of reward for each state, we split the trajectories into subsets and run MaxEnt IRL on each subset to get a unique reward function. The probability of each reward value is the proportion of the subset in the original set. Towards this end, we split the trajectories into subsets of equal sequence lengths and run MaxEnt IRL on each of them.

We compute the distribution of reward for each location as follows. Let $\mathbf{R}_s$ be a state-reward matrix. $\mathbf{R}_s$ is of dimension $l \times N$, where N is the number of states and l is the maximum sequence length. For each state S_k ($1 \leq k \leq N$), let p_k of length $n = |\mathcal{G}|$ be the vector of multinomial emission probabilities of the HMM. Let Π be the multinomial emission matrix of dimension $N \times n$ whose row vectors are p_k. We compute the location-reward matrix $\mathbf{R}_a$ as:

$$\mathbf{R}_a = \mathbf{R}_s \times \Pi. \quad (6)$$

[4] Precisely, lo_i and la_i are the sum of the coordinates of the locations weighted by the multinomial emission probabilities at S_i.

We assume that the stochastic reward $R(a)$ of each location a follows a Gaussian distribution, whose mean and variance can be derived from the corresponding column vector of $\mathbf{R}_a$.

6 Prediction

In this paper, we present two decision models to the problem of trajectory prediction: Adaptive MDP (AMDP) and Value Ratio (VR). The former follows the long-term optimal policy of an MDP and the latter uses myopic greedy heuristics to make decisions.

6.1 Adaptive MDP

Empirical evidence shows that the sequence lengths of the trajectories typically follow normal distributions [16, 18, 15]. We take advantage of this to introduce stochasticity of reward and policy into our model by splitting the training set into subsets of the same sequence lengths. For each subset, we learn a unique reward/policy function. In the end, we come up with a reward/policy matrix, where for each matrix, the columns are the states and the rows are the sequence lengths whose probability distribution follows that of the sequence lengths.

With the above setup, we obtain the following matrices from the training set for each agent type:

1. $\mathbf{R}_s$ (or $\mathbf{R}$): each entry is the reward (column) of each state that corresponds to each sequence length (row);
2. $\mathbf{V}$ ($l \times N$): each entry is the value (column) of each state that corresponds to each sequence length (row);
3. Optimal policy matrix Π^* ($l \times N$): each entry is an optimal action $a \in A$ at each state (column) that corresponds to each sequence length (row).

From $\mathbf{R}$, we are able to derive the Gaussian distribution of reward $R(s)$ at each state $s \in S$ using the probability distribution of the sequence length (i.e., the rows).

An important consideration in our model is the agent's **expected reward level**. This comes about from the observation that an agent may finish its trajectory even when there is sufficient budget to go on. Such behavior may come from an intrinsic expected reward level, such as a "personal goal", having been met. Once such goal is met, the agent would just be happy to finish there and then and not go on to maximize the cumulative reward any further. In order to model such a personal goal, we make use of the value function. From Eqn. (2), the value function at state s is sum of the immediate reward $R_{\pi(s)}(s)$ and the future expected reward. We use this future expected reward to model agent i's expected reward level Q_i:

$$Q_i = V^{\pi}(s) - R_{\pi(s)}(s). \tag{7}$$

Since both $V^{\pi}(s)$ and $R_{\pi(s)}(s)$ are given (by $\mathbf{V}$ and $R(s)$, respectively), we can derive Q_i for each agent i knowing its current state s and the current sequence length k. Furthermore, the optimal policy matrix Π^* is *stochastic* because, given a state s, each column vector of policies $\Pi^*[:, s]$ is distributed according to the Gaussian distribution of the sequence length. Algorithm 1 describes the proposed Adaptive[5] MDP decision model for trajectory prediction.

Algorithm 1 follows the long-term optimal policy of an MDP because it makes use of the optimal (stochastic) policy function to make decision at each step. The policy function is long-term optimal as a result of solving the Bellman equation (2).

[5] "Adaptive" is used to mean that the algorithm is adapted to stochastic rewards/policies and the budget constraint.

Algorithm 1 Adaptive MDP decision model for agent i

1: Given agent i's partial trajectory $\tilde{s}_i = \{(s, a)\}_i$ of current length k and i's current budget $B_i > 0$
2: Let $s = \tilde{s}_i[k]$ be the current state
3: Sample reward $R(s)$ from Gaussian distribution
4: Retrieve current state's value $\mathbf{V}[k, s]$
5: Let $Q_i = \mathbf{V}[k, s] - R(s)$ be i's expected reward level
6: Initialize i's future cumulative reward $U_i \leftarrow 0$
7: Let $\hat{s}_i \leftarrow \varnothing$ be the predicted sequence
8: **while** $U_i < Q_i$ and $B_i > 0$ **do**
9: Sample an action a from policy $\Pi^*[k : l, s]$
10: **while** $a \in \tilde{s}_i$ {a has been visited} **do**
11: Repeat Step 9
12: **end while**
13: Sample next state s' from $P_a(s, s')$
14: Update $k \leftarrow k + 1$; $s \leftarrow s'$
15: Update $\hat{s}_i \leftarrow \hat{s}_i \cup (s, a)$; $\tilde{s}_i \leftarrow \tilde{s}_i \cup \hat{s}_i$
16: Sample reward $R(s)$ from Gaussian distribution
17: Let t_a be the travel time from current location to a
18: Let Δ_a be the minimum duration to be spent at a
19: Update $U_i \leftarrow U_i + R(s)$; $B_i \leftarrow B_i - (t_a + \Delta_a)$
20: **end while**
21: Return the sequence of actions in $\hat{s}_i$

6.2 Value Ratio

At each time step, the agent samples a random reward value r_j from the Gaussian distribution $R(a_j)$ of each of the *remaining* locations a_j. Given its current location, the agent heuristically maps itself to the nearest *cluster centroid* (refer to Sect. 5.3.2) as a "point of reference" and derives the distances d_j from the cluster centroid to each of the remaining locations. The agent then chooses to visit the location j^* that has the largest ratio r_j/d_j (i.e., the ratio of the immediate reward to its cost) and repeats until its budget runs out or there is no unvisited location left. This is the well-known best "bang-for-the-buck" greedy heuristic [3]. Algorithm 2 describes the model.

Algorithm 2 Value Ratio decision model for agent i

1: Given agent i's current location a_i, its current set of *unvisited* locations $\mathcal{G}_i \subseteq \mathcal{G}$ and the current budget $B_i > 0$
2: Let $\hat{s}_i \leftarrow \varnothing$ be the predicted sequence of visits
3: **while** $|\mathcal{G}_i| > 0$ and $B_i > 0$ **do**
4: Sample reward r_j from Gaussian distribution for each $a_j \in \mathcal{G}_i$
5: Let $C_{k^*} = \arg\min_k \texttt{distance}(a_i, C_k)$ $(1 \le k \le N)$
6: Let $d_j = \texttt{distance}(a_j, C_{k^*}), \forall a_j \in \mathcal{G}_i$
7: Select a_{j^*} where $j^* = \arg\max_j r_j/d_j, \forall a_j \in \mathcal{G}_i$
8: Update $\hat{s}_i \leftarrow \hat{s}_i \cup \{a_{j^*}\}$; $\mathcal{G}_i \leftarrow \mathcal{G}_i \setminus \{a_{j^*}\}$
9: Let t_{j^*} be the travel time from a_i to a_{j^*}
10: Let Δ_{j^*} be the minimum duration to be spent at a_{j^*}
11: Update $B_i \leftarrow B_i - (t_{j^*} + \Delta_{j^*})$; $a_i \leftarrow a_{j^*}$
12: **end while**
13: Return $\hat{s}_i$

7 Experiments

7.1 Dataset

We collaborated with a large theme park operator in a major Asian city to conduct experiments and collect demographic and behavioral

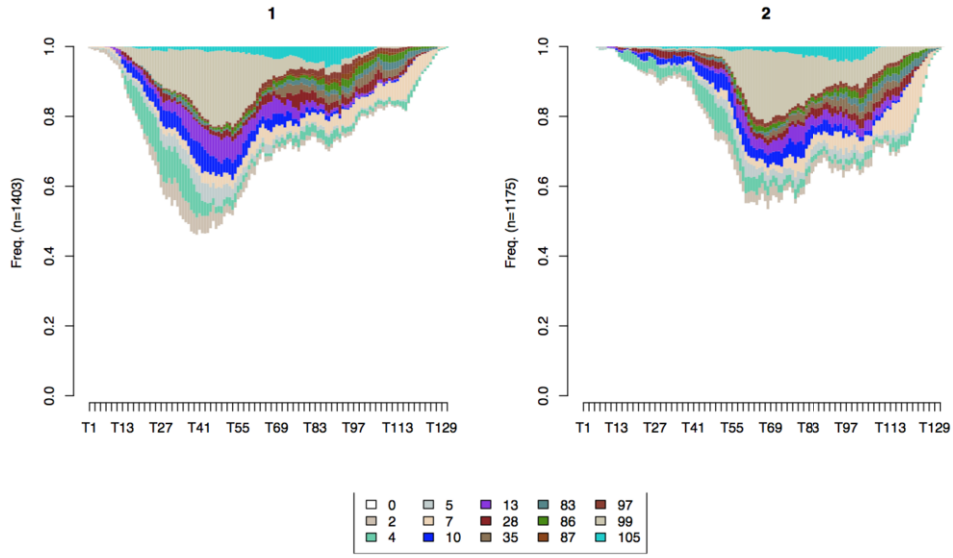


Figure 3: Visualization of the two clusters (agent types) of the training data in the experiments. Horizontal axes represent the timeline in discrete intervals of 5 minutes from 9 a.m. to 7 p.m. Vertical axes represent the probability of the visitors of each type being at each of the 14 attractions (or at some unknown location "0"). Attractions are represented by their color codes whose legend is shown at the bottom.

data from their visitors from January to April, 2014. The dataset contains the visitors' trajectories tracked using RFID devices. In the experiments, visitors pay upfront a fixed amount in order to redeem up to 14 participating attractions (locations). Visitors can only redeem the attractions during the specified 10-hour period from 9 a.m. to 7 p.m. on a chosen day. Each attraction can only be visited once.

Our dataset $\mathcal{D}$ contains trajectories of $3,867$ unique and independent visitors together with their demographic features. The empirical distribution of the sequence length of these trajectories follows a typical bell-shaped characteristic of a Gaussian distribution.

7.2 Trajectory Clustering

We perform cross-validations[6] on $\mathcal{D}$. For each fold, the training set $\mathcal{S}$ is used for trajectory clustering and decision modeling. Our hierarchical clustering results in $K = 2$ clusters using the interval $\Delta_\tau = 5$ minutes (refer to Sect. 5.1) for all the agents. The value of K was chosen based on inspection of the hierarchical tree and empirical goodness of clustering via the silhouette coefficient (partitions of comparable sizes and good in-group cohesiveness).

Fig. 3 visualizes the 2 clusters using training data of one of the random folds. The horizontal axes represent the discretized timeline (by Δ_τ) from 9 a.m. to 7 p.m. for each cluster and the vertical axis represents the probability for each agent of each cluster to be at any one of the 14 attractions at any interval. (Note that even after 7 p.m., some activities can still be recorded in the park.) The attractions are identified by their numbers whose color codes are shown in the legend at the bottom of the figure. We denote "0" (white color) when we do not know for sure the location of an agent during a given time interval (i.e., he was not observed at any known attraction during the interval). We can see that, most of the time, visitors hang out in the park without checking into any specific attractions.

The trajectory clustering reveals that the main differences between the two agent types are their temporal behaviors. That is, agent type 1 tends to arrive earlier and has their peak of visiting activities earlier in the day (around 12–1 p.m.), and then (their visit frequency) sharply drops off. Whereas, agent type 2 tends to arrive much later and reaches their peak later (at round 3–4 p.m.), and then gradually declines. If budget is defined as the duration from the time of entry until the closing time (7 p.m.), then agent type 1 has, on average, 114 minutes more than agent type 2. As a result, we call agent type 1 the "early birds" and agent type 2 the "latecomers". The two clusters have roughly comparable sizes with cluster 1 being 54.42% and cluster 2 being 45.58% of the set training $\mathcal{S}$.

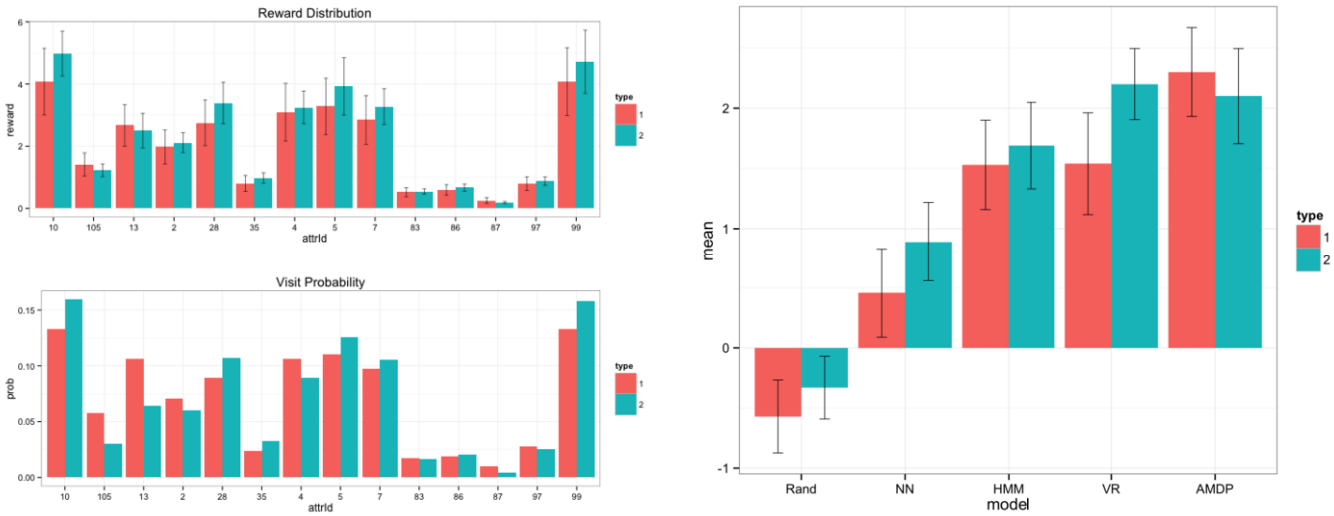
[6] Precisely, we performed 3-fold cross-validations to ensure a large enough training/test partition per fold.

7.3 Evaluation

For each cluster in $\mathcal{S}$, we learn the matrices $\mathbf{R}$, $\mathbf{V}$, and Π^*. The test set $\mathcal{T}$ is used to validate the predicted trajectories. For each agent $i \in T$, let l_i be i's final sequence length. We first predict i's type using its demographic features and first timestamp via a multinomial logistic model. Given i's partial trajectory of length k, we predict i's remaining trajectory while varying $k \in [2, l_i - 1]$. Let s_i^* and $\hat{s}_i$ be i's actual and predicted remaining trajectory, respectively. We use the Levenshtein edit distance [6] to quantify the similarity between s_i^* and $\hat{s}_i$. Each match receives a fixed positive score and each mismatch incurs a negative penalty proportional to the distance (in kilometers) between the two locations.

The following baseline models are used for evaluation:

1. **HMM.** At each time step, predict agent i's current state s, generate an *unvisited* location based on the state's multinomial probabilities p_s and repeat until B_i runs out. This is based on [20].
2. **Nearest neighbor.** At each time step, agent i redeems a remaining location that is nearest to its current location and repeats until B_i runs out.
3. **Random.** At each time step, i redeems a random unvisited location and repeats until B_i runs out.



(a) (b)

Figure 4: (a) Comparison between the estimated reward distribution for each attraction ("attrId") (top panel) and the empirical visit probability for each attraction (bottom panel). (b) Similarity measures between the actual and predicted trajectories across different models: Random ("Rand"), Nearest Neighbor ("NN"), HMM, Value Ratio ("VR") and Adaptive MDP ("AMDP").

7.4 Results

Our experimental results are summarized in Fig. 4. In Fig. 4a, the mean reward per attraction learned from IRL and Eqn. (6) is plotted together with its 95% confidence interval (top panel). The figure shows that the mean rewards, in general, faithfully reflect their respective empirical probabilities of attraction visit for both agent types (i.e., their preferences – in the bottom panel).

It is noteworthy to observe in Fig. 4a that agent type 2 has, for the most part, higher (absolute) *immediate* reward per attraction (top panel) than agent type 1. This consequentially differentiates the underlying decision processes employed by the two agent types. Fig. 4b shows the distributions of the similarity measures (means and variances – represented by 95% confidence bars) across the models. Each distribution is computed from the cross-validation while varying the observed partial trajectory length $k \in [2, l_i - 1]$. A higher mean similarity implies a more accurate prediction, on average. These distributions (in Fig. 4b) are empirically verified to be Gaussian.

For agent type 1, Fig. 4b shows that the Adaptive MDP model has the most accurate prediction, on average. The Value Ratio and HMM model both have about the same second best average prediction score. The Random baseline model has the least accurate average prediction, which is quite reasonable, followed by the Nearest Neighbor model. For agent type 2, the figure shows that the Adaptive MDP model performs marginally worse than the Value Ratio model, even though it still fares much better than the other baselines. In other words, the Value Ratio model makes the most accurate prediction, on average, for this group of agents. This is a remarkable result that warrants further discussion.

7.5 Discussion

From trajectory clustering, we have discovered that agent type 1 are the early birds and agent type 2 are the latecomers. From the perspective of modeling, agent type 1 has a much larger budget (by 114 minutes, on average) than agent type 2. Larger budget means more flexibility, more foresight and better long-term planning, which is what the Adaptive MDP model reflects: it embodies the long-term optimal policy of the corresponding reinforcement learning model. This indeed performs better than other short-sighted baselines.

On the other hand, a smaller budget, which agent type 2 has, translates into less flexibility and less time for careful planning, which ultimately results in more myopic and suboptimal decisions (i.e., resorting to greedy strategies). This is reflected in the experimental results, where the greedy and myopic Value Ratio model performs the best for agent type 2 (even though just marginally better than the Adaptive MDP). This myopic decision-making corroborates with the observations in Fig. 4a, where most of agent type 2's immediate rewards are larger (in absolute terms) than agent type 1's such that it sees less values in *delayed* (future) rewards and finds more incentives to act greedily [24]. This is also evidenced in Fig. 1, where type 2 has a much stronger tendency to visit attractions that are nearby to one another (i.e., maximizing the value ratio) than type 1.

8 Conclusion

In this paper, we address the problem of trajectory prediction using reinforcement learning to model the agent's sequential decisions. By doing so, we have discovered from real-world trajectories how people make decisions: they make more optimal decisions when given enough time to do so. This is perhaps not surprising in retrospect, because it is reasonable that foresighted decisions and careful plans need time to coordinate, while myopic ones do not (as only the immediate rewards are considered). On the other hand, this also validates our framework's ability to model real-world behaviors by finding out what makes reasonable sense in real life.

Our main shortcoming here is the simplistic handling of the budget constraint. We would like to see if handling it in more sophisticated ways would improve predictions. For example, for foresighted agents, we would like to experiment with decision models other than MDP in our future work. One of which is the adaptive stochastic knapsack [13], which it is similar to a traditional knapsack model ex-

cept for the sequential decisions and stochastic reward of each item. Another shortcoming of this work is the simplistic Value Ratio model for myopic decision-making (type 2), which yields just slightly better prediction than the Adaptive MDP for agent type 2. Hence, for myopic agents, a more sophisticated decision model may be desirable to better model and predict their behaviors. One of such model for sequential decisions has been proposed in the operations research literature [23]. This is also worth investigating in the future work.

ACKNOWLEDGEMENTS

This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Program Office, Media Development Authority, as well as its Corp Lab @ University scheme.

Siyuan Liu is additionally supported by the Basic Research Program of Shenzhen: JCYJ20140610152828686 and the Natural Science Foundation of China: 61572488.

REFERENCES

[1] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y. Ng, 'An application of reinforcement learning to aerobatic helicopter flight', in *Advances in Neural Information Processing Systems*, eds., B. Schlkopf, J. Platt, and T. Hoffman, volume 19, Cambridge, MA, USA, (2007). MIT Press.

[2] Pieter Abbeel and Andrew Y Ng, 'Apprenticeship learning via inverse reinforcement learning', in *Proceedings of the Twenty-first International Conference on Machine Learning*, p. 1. ACM, (2004).

[3] Maria-Florina Balcan, Amit Daniely, Ruta Mehta, Ruth Urner, and Vijay V Vazirani, 'Learning economic parameters from revealed preferences', in *Web and Internet Economics*, 338–353, Springer, (2014).

[4] Leonard E Baum, Ted Petrie, George Soules, and Norman Weiss, 'A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains', *The Annals of Mathematical Statistics*, **41**(1), 164–171, (1970).

[5] R. Bellman, 'A Markovian decision process', *Journal of Mathematics and Mechanics*, **6**(4), 679–684, (April 1957).

[6] Paul E Black, 'Levenshtein distance', *Algorithms and Theory of Computation Handbook*, (1999).

[7] G David Forney Jr, 'The Viterbi algorithm', *Proceedings of the IEEE*, **61**(3), 268–278, (1973).

[8] Chris Fraley and Adrian E Raftery, 'Model-based clustering, discriminant analysis, and density estimation', *Journal of the American Statistical Association*, **97**(458), 611–631, (2002).

[9] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez, 'Next place prediction using mobility Markov chains', in *Proceedings of the First Workshop on Measurement, Privacy, and Mobility*, p. 3. ACM, (2012).

[10] Gerd Gigerenzer, Reinhard Selten, et al., 'Rethinking rationality', *Bounded rationality: The adaptive toolbox*, 1–12, (2001).

[11] Valerie Guralnik and Karen Zita Haigh, 'Learning models of human behaviour with sequential patterns', in *Proceedings of the AAAI-02 Workshop on Automation as Caregiver*, pp. 24–30, (2002).

[12] De-An Huang, Amir-massoud Farahmand, Kris M Kitani, and J Andrew Bagnell, 'Approximate maxent inverse optimal control and its application for mental simulation of human interactions', in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, (2015).

[13] Taylan Ilhan, Seyed MR Iravani, and Mark S Daskin, 'The adaptive knapsack problem with stochastic rewards', *Operations Research*, **59**(1), 242–248, (2011).

[14] John Krumm and Eric Horvitz, 'Predestination: Inferring destinations from partial trajectories', in *UbiComp 2006: Ubiquitous Computing*, 243–260, Springer, (2006).

[15] Truc Viet Le, Siyuan Liu, and Hoong Chuin Lau, 'Reinforcement learning framework for modeling spatial sequential decisions under uncertainty', in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pp. 1449–1450. International Foundation for Autonomous Agents and Multiagent Systems, (2016).

[16] Truc Viet Le, Siyuan Liu, Hoong Chuin Lau, and Ramayya Krishnan, 'A quantitative analysis of decision process in social groups using human trajectories', in *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pp. 1425–1426. International Foundation for Autonomous Agents and Multiagent Systems, (2014).

[17] Truc Viet Le, Siyuan Liu, Hoong Chuin Lau, and Ramayya Krishnan, 'Predicting bundles of spatial locations from learning revealed preference data', in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 1121–1129. International Foundation for Autonomous Agents and Multiagent Systems, (2015).

[18] Siyuan Liu, Qiang Qu, and Shuhui Wang, 'Rationality analytics from trajectories', *ACM Transactions on Knowledge Discovery from Data (TKDD)*, **10**(1), 10, (2015).

[19] Haiping Ma, Huanhuan Cao, Qiang Yang, Enhong Chen, and Jilei Tian, 'A habit mining approach for discovering similar mobile users', in *Proceedings of the 21st International Conference on World Wide Web*, pp. 231–240. ACM, (2012).

[20] Wesley Mathew, Ruben Raposo, and Bruno Martins, 'Predicting future locations with hidden Markov models', in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pp. 911–918. ACM, (2012).

[21] Andrew Y Ng and Stuart Russell, 'Algorithms for inverse reinforcement learning', in *Proc. 17th International Conf. on Machine Learning*, (2000).

[22] Stuart Russell, 'Learning agents for uncertain environments (extended abstract)', in *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pp. 101–103, New York, NY, USA, (1998). ACM.

[23] Matthew J Sobel and Wei Wei, 'Myopic solutions of homogeneous sequential decision processes', *Operations Research*, **58**(4-part-2), 1235–1246, (2010).

[24] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, USA, 1998.

[25] Shao Zhifei and Er Meng Joo, 'A review of inverse reinforcement learning theory and recent advances', *International Journal of Intelligent Computing and Cybernetics*, **5**(3), 293–311, (June 2012).

[26] Brian D. Ziebart, Anind K. Dey, and J. Andrew Bagnell, 'Probabilistic pointing target prediction via inverse optimal control', in *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces*, pp. 1–10, New York, NY, USA, (February 2012). ACM.

[27] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey, 'Maximum entropy inverse reinforcement learning.', in *AAAI*, pp. 1433–1438, (2008).

[28] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey, 'Human behavior modeling with maximum entropy inverse optimal control.', in *AAAI Spring Symposium: Human Behavior Modeling*, p. 92, (2009).